

FAST COMPUTATION OF ARRIVAL TIMES IN HETEROGENEOUS MEDIA

I. BERRE^A, K. H. KARLSEN^A, K.-A. LIE^B, AND J. R. NATVIG^B

ABSTRACT. We continue the work that was initiated in [7] on a fast marching-like method for simulating two-phase incompressible immiscible flow of water and oil in a porous medium. The purpose of the present paper is threefold: (1) To provide an alternative derivation of the fast-marching-like method which reveals a strong connection to modern streamline methods. (2) To investigate the capabilities of this method as a tool for simulating realistic two-phase flow scenarios. In particular, we discuss, and demonstrate via numerical examples, two problems in this respect. First, the fast-marching-like algorithm does not always compute the correct solution. Second, the upwind difference scheme on which the method is based is inaccurate for flows parallel to high-contrast layers in the permeability field. (3) To present possible improvements of the work in [7] in order to deal with some of the concerns raised herein.

1. INTRODUCTION

The mathematical model for porous media flow considered herein describes the flow of two inviscid, incompressible and immiscible fluids in a domain $\Omega \subset \mathbf{R}^3$ filled with a porous medium. In reservoir simulation, these fluids are typically oil and water. Neglecting capillary pressure and gravity, the motion of the two fluids is governed by the a scalar conservation law

$$(1) \quad \frac{\partial S}{\partial t} + \mathbf{v}(\mathbf{x}) \cdot \nabla f(S) = 0,$$

where S is the saturation of water and f is the fractional flow function accounting for different flow resistance of the two fluids. A complete model must also include an equation for the fluid velocity \mathbf{v} . The most commonly used model is a constitutive relation known as Darcy's law that relates \mathbf{v} to the fluid pressure. The continuity equation can then be stated as

$$(2) \quad \nabla \cdot (\lambda_T \nabla P) = 0 \text{ in } \Omega,$$

$$(3) \quad \mathbf{v} = -\lambda_T \nabla P,$$

where P is the total fluid pressure and λ_T is the total mobility which depends on the absolute permeability as well as on S . We model injection and production of fluid with Neumann boundary conditions on $\Gamma^+ = \{\mathbf{v} \cdot \mathbf{n} > 0\}$ and $\Gamma^- = \{\mathbf{v} \cdot \mathbf{n} < 0\}$ respectively. The equations (2)–(3) yield a divergence free irrotational velocity field. We refer to [1, 3] for more general models.

The model (1)–(3) is solved most effectively by applying to each equation a specialized numerical method, which accurately resolves important features of the equation, see for example [5] and references therein. Recently, Karlsen, Lie and Risebro [7] proposed a new numerical method for the solution of (1) for a fixed velocity field \mathbf{v} . Their approach was an attempt to apply the level-set technology of Osher and Sethian [10], together with the fast-marching method [13, 16, 17, 6], to the propagation of curves of constant water saturation $S = \sigma$. In the absence of shocks, each point on these level curves moves with a characteristic speed $f'(\sigma)\mathbf{v}$. Thus, in this case the tracking of a saturation contour for a fixed saturation σ can be recast as the tracking of the zero level set of a function ϕ that is described by the PDE

$$(4) \quad \frac{\partial \phi}{\partial t} + f'(\sigma)\mathbf{v} \cdot \nabla \phi = 0.$$

^A Department of Mathematics, University of Bergen, Johs. Brunsgt. 12, N-5008 Bergen, Norway.
 E-mail: inga.berre@mi.uib.no, kenneth.karlsen@mi.uib.no.

^B SINTEF Applied Mathematics, P.O. Box 124 Blindern, N-0314 Oslo, Norway.
 E-mail: Knut-Andreas.Lie@sintef.no, Jostein.R.Natvig@sintef.no.

In general, the solution obtained by tracking all saturation contours according to (4) will lead to a multivalued solution. To compute the entropy solution of (1) we must modify the wave speed $f'(\sigma)\mathbf{v}$ when shocks form, i.e., when level sets for different σ coincide. To avoid this complication we henceforth assume that the relative speeds of the level sets are monotone in σ . A simple way to ensure this is to consider only initial saturations formed by two constant states, say σ^+ in \mathcal{W} and σ^- in $\Omega \setminus \mathcal{W}$, separated by a contour at $\partial\mathcal{W}$. Then, all level sets of the saturation initially coincide with $\partial\mathcal{W}$. For this Riemann-like initial value problem, the relative speed of each contour is constant: If $\sigma^+ > \sigma^-$ (or $\sigma^+ < \sigma^-$), the relative speed is given as the derivative of the upper convex (or lower concave) envelope f_c of f between σ^- and σ^+ . The single valued solution of (1) in $\Omega \setminus \mathcal{W}$ can then be obtained by solving

$$(5) \quad \frac{\partial\phi}{\partial t} + f'_c(\sigma)\mathbf{v} \cdot \nabla\phi = 0$$

for all saturation contours. From a physical point of view, this corresponds to injection of a fixed liquid mixture into a reservoir containing another homogeneous fluid mixture. Note that we must have $\Gamma^+ \subseteq \mathcal{W}$ since fluid must flow out of \mathcal{W} . From here on, we will simply use $\mathcal{W} = \partial\mathcal{W} = \Gamma^+$.

Since the motion of the level sets are identical up to a constant in the wave speed, we should be able to compute them all in one go. If we require that $\mathbf{v} \cdot \mathbf{n} > 0$ on the initial contour $\partial\mathcal{W} = \Gamma^+$, a level set will not pass any point in the domain more than once. Then we can set $\phi(\mathbf{x}, t) = T(\mathbf{x})/f'_c(\sigma) - t$ for some function T independent of σ , and (5) is reduced to a linear boundary-value problem

$$(6) \quad \mathbf{v} \cdot \nabla T = 1 \text{ in } \Omega, \quad T = 0 \text{ on } \Gamma^+.$$

In this case T represents the motion of a level set with unit relative speed. Thus, we can interpret $T(\mathbf{x})$ as the time needed for a passive test particle to travel from some point on Γ^+ to \mathbf{x} . We will therefore denote this quantity by arrival time or time-of-flight and (6) as the arrival-time equation. Ultimately, it is the saturation that is to be computed. We recover the saturation by asking which saturation level set has reached a given point \mathbf{x} at time t ; that is, by solving $0 = T(\mathbf{x})/f'_c(\sigma) - t$ for σ . This is the well known Buckley-Leverett profile, see [7] for more details. With this approach, the amount of work has been reduced to the solution of the boundary value problem (6) and the inversion of a function f'_c (which can be done analytically). In [7], (6) was solved using a fast-marching-like algorithm that has an asymptotic cost of $\mathcal{O}(N \log N)$ operations for N unknowns. While the numerical examples in [7] were two-dimensional, some three-dimensional examples with the same method were presented in [8]. In [2] the method was extended to incorporate capillary forces (i.e., viscous terms in the transport equation) through the use of the transport-collapse operator.

In this paper we will first give an alternative derivation of the fast-marching-like method presented in [7] that will demonstrate a close connection between this approach and that of modern streamline methods. For an overview of the research on streamlines methods, we refer to [9] and the references therein. We then continue with an investigation of the fast-marching-like method. In [7, 8], numerical tests showed that the method was fast and reliable and that the accuracy was comparable to some other methods. Unfortunately, the formulation of a boundary value problem for the arrival time and the subsequent fast-marching-like solution procedure is not as robust as first reported. A more detailed analysis of the discretization and the solution procedures reveals two problems. These problems are not apparent in the presentation of [7]. The first problem is related to the discretization of (6) for discontinuous media. The second problem is that the discretization of (6) lacks the causality principle that the fast-marching method hinges on. The purpose of this paper is to point out and discuss these problems and to present possible improvements of the method in [7] so as to avoid (or reduce) these defects. Since these problems are related to (6), we will in the rest of this paper focus attention on arrival times and not on the computation of saturations. In addition, since the approach taken here is readily extended to three spatial dimensions, we present examples in two spatial dimensions only.

The outline of the paper is as follows: In Section 2 we describe the concept of time-of-flight and how this can be used to split the saturation equation in two simpler problems. Then, in Section 2.1

we present the discretization of the arrival-time equation and demonstrate its shortcomings. In Section 2.2 we give a brief outline of the classical fast-marching method and explain why the fast-marching-like method of [7] does not perform well for strongly varying velocity fields. In Section 3 we propose an alternative fast(er) solution method for the discretised equations that does not suffer from the same deficiency as the fast-marching-like algorithm employed in [7]. In Section 3.2, we propose a way to reduce the large discretisation error we have observed. Finally, Section 4 is devoted to discussions relating our numerical approach to modern streamline methods.

2. REFORMULATION OF THE SATURATION EQUATION

In this section we present a slightly different perspective on the numerical method in [7]. While this method was motivated as a level-set formulation of the saturation equation in [7], we will see the method here in the form of modern streamline methods. This form is based on the concept of time-of-flight or arrival time. The time-of-flight formalism [4] has made a profound impact upon the development of modern streamline methods. (See [9] for an overview of streamline methods). As we will see later, this formalism is also central to the understanding of the approach taken in [7]. We will therefore introduce streamlines and the time-of-flight formalism in some detail.

A streamline Ψ is the path traced out by a passive test particle moving with the flow given by a forcing velocity field \mathbf{v} such that the vector \mathbf{v} is tangential to Ψ at every point. Along each streamline one can reparametrize the space coordinate by introducing the travel time, which is commonly referred to as the *time-of-flight*. In [7] this quantity is referred to as arrival time. The time-of-flight $T(\mathbf{x})$ measures the time a passively advected test particle needs to travel along a streamline from its initial position on Γ^+ to a given point \mathbf{x} . This travel time can be defined by the following integral along a streamline Ψ ,

$$(7) \quad T(\mathbf{x}) = \int_{\Psi} \frac{ds}{|\mathbf{v}(\mathbf{x}(s))|}.$$

Thus, for our two-phase model (1)–(3), the arrival time gives a picture of the forcing velocity field, while not taking into account the nonlinear effects of f caused by the relative mobilities of water and oil. Alternatively, the arrival time is given by the differential equation (6) (see [7, 9])

$$\mathbf{v} \cdot \nabla T = 1 \text{ in } \Omega, \quad T = 0 \text{ on } \Gamma^+.$$

To derive this equation, we can consider an infinitesimal movement of the test particle. If s denotes the distance in the direction $\mathbf{n} = \mathbf{v}/|\mathbf{v}|$, we have by definition that $\partial s / \partial T = |\mathbf{v}|$, or in other words

$$1 = |\mathbf{v}| \frac{\partial T}{\partial s} = |\mathbf{v}| \left(\nabla T \cdot \frac{\mathbf{v}}{|\mathbf{v}|} \right) = \nabla T \cdot \mathbf{v}.$$

Changing perspective, $T(\cdot)$ maps each level set of T to a positive real number. Thus, T can be understood as the coordinate mapping that transforms (1) into a family of one-dimensional conservation laws, where T takes the role of the space coordinate. To see this, let T be the time-of-flight. If we consider the saturation equation along a streamline for \mathbf{v} , it takes the form

$$\frac{\partial \sigma}{\partial t} + |\mathbf{v}| \frac{\partial f(\sigma)}{\partial s} = 0.$$

Then we can substitute the equation for the time-of-flight along a streamline to get

$$(8) \quad \frac{\partial \sigma}{\partial t} + \frac{\partial f(\sigma)}{\partial T} = 0.$$

The data for these one-dimensional problems are derived from the data for (1). If we supply the following initial and boundary values for (1):

$$\begin{aligned} S(\mathbf{x}, 0) &= S_0(\mathbf{x}) \text{ in } \Omega, \\ S(\mathbf{x}, t) &= S_D(\mathbf{x}, t) \text{ on } \Gamma^+, \end{aligned}$$

the corresponding data for (8) are

$$\begin{aligned}\sigma(T, 0) &= \sigma_0(T) = S_0(\Psi(T(\mathbf{x}))), \\ \sigma(0, t) &= \sigma_D(t) = S_D(\Psi(0), t).\end{aligned}$$

The method in [7] computes the map $T(\mathbf{x})$ for all \mathbf{x} but does not solve for individual streamlines Ψ . In other words, the method does not explicitly associate a unique flow-path with each \mathbf{x} . Therefore all streamlines are treated as one when computing the saturation, and we must require that the data σ_0 and σ_D are the same for all Ψ . This is a slight generalization from [7], where the data was restricted to constant injected fluid mixture σ_D and constant initial saturation $\sigma_0(t) = \sigma_0$ in all of Ω . In the present formulation it is evident that we can solve any initial-value problem that has equal data σ_D and σ_0 for all streamlines. For a level-set based approach to more general initial-value problems for the saturation equation we refer to [2].

2.1. The discrete arrival-time equation. To solve the boundary value problem (6) for the arrival time T , an upwind discretization for (6) can be constructed once a fluid velocity \mathbf{v} is given. In two spatial dimensions the main discretization used in [7, 8] can be constructed as follows. Let the domain be regularly partitioned into grid blocks of dimensions Δx and Δy and let the nodes for arrival time be located in the center of each grid block. Assume that we trace a streamline Ψ from the center \mathbf{x}_k of block k to a point \mathbf{y} in the upwind direction $-\mathbf{v}$. We can call the segment of the streamline from \mathbf{x}_k to \mathbf{y} for ψ_k . The upwind discretization of (6) can then be written as

$$(9) \quad T_k = T(\mathbf{x}_k) = T(\mathbf{y}) + \Delta T_k,$$

$$(10) \quad \Delta T_k = \int_{\psi_k} \frac{ds}{|\mathbf{v}(\mathbf{x}(s))|}.$$

To complete the scheme, we must specify how to evaluate the integral (10) and how to compute $T(\mathbf{y})$. We approximate the streamline segment ψ_k by a straight ray from \mathbf{x}_k in the $-\mathbf{v}$ direction, and we assume \mathbf{v} is constant along this ray. Thus, $\Delta T_k = |\mathbf{x}_k - \mathbf{y}|/|\mathbf{v}|$. The point \mathbf{y} is assumed to lie on the straight line connecting two neighboring nodes \mathbf{x}_i and \mathbf{x}_j of \mathbf{x}_k and is therefore given as $\mathbf{y} = \alpha \mathbf{x}_i + (1 - \alpha) \mathbf{x}_j$ for some scalar $\alpha \in [0, 1]$. To compute $T(\mathbf{y})$ we use a simple linear interpolation from the nodes i and j near \mathbf{y} ,

$$(11) \quad T_k = \alpha T_i + (1 - \alpha) T_j + \Delta T_k,$$

where T_i and T_j are the arrival times of node i and j , respectively. In three dimensions we use a similar bilinear interpolation in the triangle spanned by three neighboring nodes in one of the axial planes.

The upwind discretization gives a consistent way to compute the arrival time in the whole domain. The difference scheme can be written as a linear system of equations

$$(12) \quad \sum_j A_{ij} T_j = \Delta T_i, \quad i = 1 \dots N,$$

where $A = \{A_{ij}\}$ contains the coefficients of (11). Before we take a closer look at the fast-marching-like algorithm from [7], we will examine how the difference scheme compares to a direct evaluation of (7). To evaluate the integral in (7), we use a standard approximate algorithm for integrating streamlines, as first reported by Pollock [12].

We present three numerical examples: one with a smooth velocity field, one with a mildly heterogeneous velocity field, and one with a non-smooth velocity field. To simplify the presentation, the examples are in two spatial dimensions. In all examples, the solution of the linear system (12) is computed using an iterative solver. To have a measure of the discretization error associated with (11), we compute the residual

$$(13) \quad r = |AT_a - \Delta T|,$$

where T_a is an approximation of (7) evaluated in each node in the grid. This will give us an idea of the accuracy we can expect from the difference scheme (9), which was the basis for the fast-marching-like method in [7].

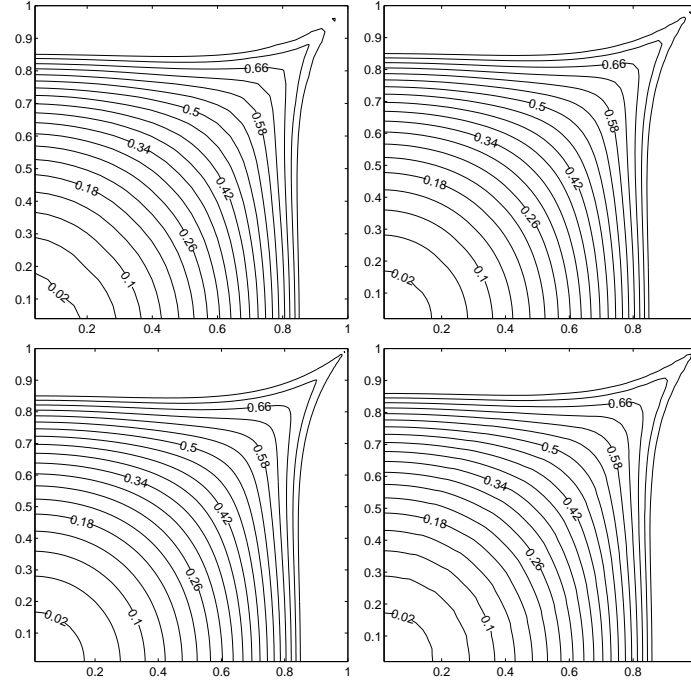


FIGURE 1. Convergence study for the homogeneous quarter-five spot test case with mesh spacing $h = 0.04$ (upper left), $h = 0.02$ (upper right), and $h = 0.01$ (lower left). The lower left plot shows the arrival time computed by a direct evaluation of (7) with $h = 0.02$. Contours are only drawn for $T \leq 0.74$.

Example 1. To demonstrate that the scheme (12) works well for smoothly varying velocity fields \mathbf{v} , we first compute T for a homogeneous quarter-five spot test case. We set $\Omega = [0, 1] \times [0, 1]$ with no-flow boundaries. In the lower left corner we place an injector and in the upper right-hand corner we place a producer. These are modeled as point sources with intensity $+1.0$ and -1.0 respectively. This means that one pore volume of fluid is injected per time unit. The corresponding data for (6) is a homogeneous Dirichlet boundary Γ^+ around the injector. The velocity field \mathbf{v} is computed using Darcy's law (3). To discretise (6) we have used mesh spacing $\Delta x = \Delta y = h$. Figure 1 shows the solution of the linear system (12) for three different grid resolutions: $h = 0.04$, $h = 0.02$, and $h = 0.01$. For comparison, we have also computed the time-of-flight by direct evaluation of (7). The streamlines Ψ have been approximated using Pollock's method [12] with mesh spacing $h = 0.02$. As we can see from the right column in Figure 1, the two solution strategies are almost visually indistinguishable as expected. From the three plots, we observe that the discretization (11) seems to converge as the grid is refined. In Figure 5 (left) we have plotted the discretisation error (13) for this example. As we can see, the error is small throughout the domain. \square

Example 2. We repeat the problem of Example 1 with the heterogeneous permeability field shown in Figure 2. The permeability varies between 40mD and 15D. The time-of-flight for these problems are plotted in Figure 3. These solutions are compared to an approximation of (7), also shown in Figure 3. Again, we observe that the scheme (11) seems to converge as the grid is refined. However, the convergence is a bit slower than in Example 1. Again, we have plotted the discretisation error (13) in Figure 5 (middle). The error is clearly larger for this example, but still of order h . \square

Example 3. We now compute Example 5.3 from [7]. This example describes flow in a porous cross-shaped channel defined on $\Omega = (0, 1) \times (0, 1)$. The permeability is set equal to 100mD in the beams of the cross and 0.01mD outside; injectors are placed in the lower and left-hand beams, and the producers are placed in the upper and right-hand beams. This flow field is almost discontinuous along the walls of the channel and smooth inside. We compute T on a uniform

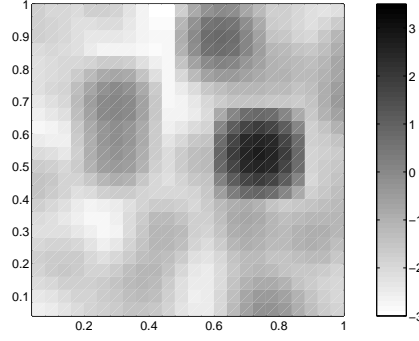


FIGURE 2. Logarithm of the permeability field for Example 2. The permeability is given on a grid with $h = 0.04$. This permeability field is used for the finer grids as well.

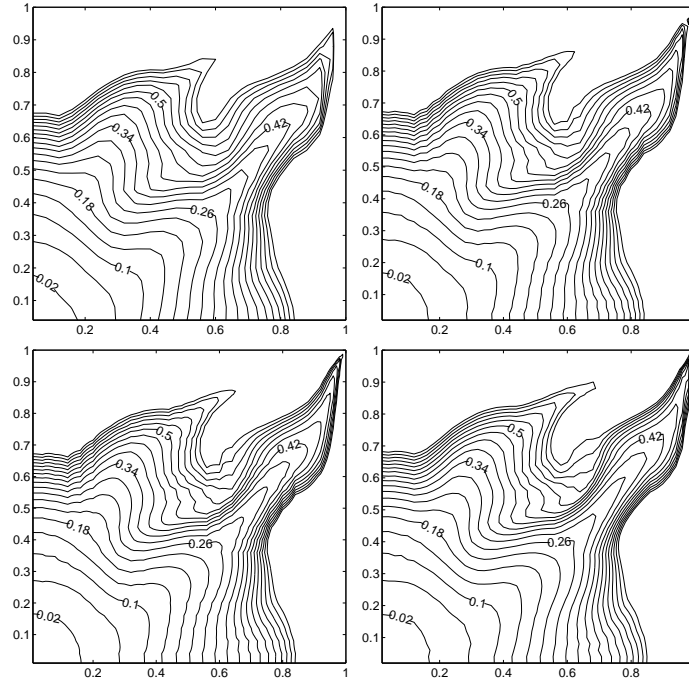


FIGURE 3. Convergence study for the homogeneous quarter-five spot test case with $h = 0.04$ (upper left), $h = 0.02$ (upper right), and $h = 0.01$ (lower left). The lower left plot shows the arrival time computed by a direct evaluation of (7) with $h = 0.02$. Contours are only drawn for $T \leq 0.70$.

grids with spacing $h = 0.02$, $h = 0.01$ and $h = 0.005$. We solve the system (12) for the different discretizations and compare the results with an approximation to (7) as before. Figure 4 shows contour plots of the computed time-of-flights. As we can see, the solution computed using the difference scheme is reasonably accurate in some parts of the channel, and wrong in others. In the plot of the error (13) in Figure 5, we can clearly see that the discretisation error is small except in layers of grid blocks at the walls of the channel. These layers coincide with the part of the wall where the difference formula (11) includes a node both inside and outside the channel, or in other words, where the difference scheme has a stencil that covers a region where T has a large jump! Thus the discretisation error increases from $\mathcal{O}(h)$ to $\mathcal{O}(1)$ in a layer of cells along the wall. Therefore the large discrepancy between the difference scheme and the integral computation

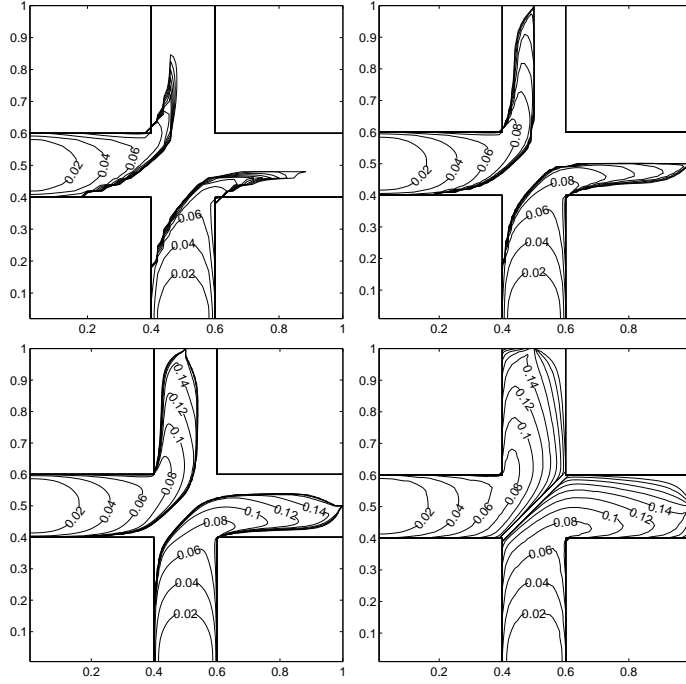


FIGURE 4. Convergence study for the channel problem with $h = 0.02$ (upper left), $h = 0.01$ (upper right), and $h = 0.005$ (lower left). (lower right) The lower left plot shows the arrival time computed by a direct evaluation of (7) with $h = 0.01$. Contours are only drawn for $T \leq 0.24$.

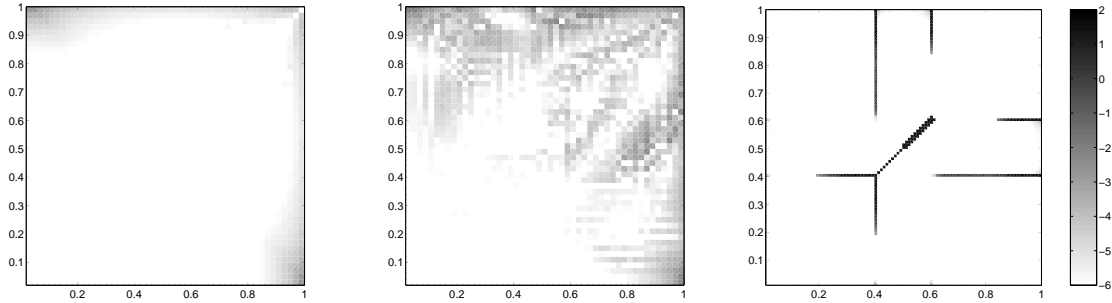


FIGURE 5. Logarithm of residual (13), $\log r$, for the three test cases. (left) Homogeneous case with $h = 0.02$; (middle) heterogeneous case with $h = 0.02$; and (right) channel problem with $h = 0.01$. Note that the residual is only plotted inside the channel.

of T is caused by the discretization errors along the walls, and these errors are convected in the direction of the flow. \square

Examples 1 to 3 demonstrate that while the difference scheme is accurate for smooth data, it has low accuracy for high contrast layered permeabilities where the fluid velocity is nearly parallel to the layers. In this situation there is a large shear in the velocity that makes interpolation of time-of-flight between adjacent streamlines inaccurate. In addition, any errors that occur in the computation accumulate in the direction of flow. Thus the time-of-flight will be more and more smeared out downstream. Note also that if these arrival times were to be used to compute saturations, the resulting solution would have a large mass balance error. Example 3 therefore demonstrates that for certain boundary value problems with large jumps in the velocity, the method (12) is inferior to methods that approximate (7) directly. A fast method based on

reordering of unknowns will, if consistent, give no better results than a direct solution of (12), for the discretization (9).

At the time of publication of [7, 8] these problems were unknown to the authors. In fact, the early results obtained by the new method looked very promising. We will come back to the reason why the problems went unnoticed at the end of the next section, where we discuss another fundamental problem with the approach in [7].

2.2. The fast-marching method. In [7] the difference scheme (11) was solved using a fast-marching-like method. In this section, we explain why the fast-marching-like method may be an inappropriate method for solving the arrival-time equation (6).

Originally, the fast-marching method [13, 16, 17, 6] was developed as an efficient method to compute the solution of the Eikonal equation,

$$(14) \quad |\nabla\phi| = \frac{1}{c(\mathbf{x})}, \text{ in } \Omega \setminus \Omega_0, \quad \phi = 0 \text{ on } \partial\Omega_0,$$

where Ω_0 is some reasonable subset of Ω and $c(\mathbf{x})$ is the local wave speed. The Eikonal equation is the first-order term in a geometrical-optics approximation to the wave equation. It characterizes the motion of wave fronts originating from $\partial\Omega_0$. At any time $t \geq 0$, the shape and position of the wave front is given by the contour $\{\phi = t\}$. Equation (14) states that the wave front propagates in its normal direction with the spatially varying speed c . The front may develop corners and cusps even for smooth c and $\partial\Omega_0$. The characteristics of the Eikonal equation are perpendicular to the level sets of ϕ , and ϕ is strictly increasing along the characteristics. Thus, consider a characteristic ψ within the simplex spanned by three points, say \mathbf{x}_{ij} , $\mathbf{x}_{i-1,j}$, $\mathbf{x}_{i,j+1}$. If the wave speed c is constant within the simplex, then ψ will be a straight line. It follows that if ψ passes through \mathbf{x}_{ij} and ϕ is increasing along ψ toward \mathbf{x}_{ij} , then

$$(15) \quad \phi_{ij} \geq \max(\phi_{i-1,j}, \phi_{i,j+1}).$$

This property is referred to as *causality* in the fast-marching terminology, and depends only on the assumption that each such simplex in the grid has no obtuse angles. To solve equation (14) on a Cartesian grid $\Lambda = \{i\Delta x, j\Delta y\}$ we set $\phi_{ij} = \phi(i\Delta x, j\Delta y)$ and use the following first-order upwind-discretization

$$(16) \quad \sqrt{\max(D_{ij}^{-x}\phi, -D_{ij}^{+x}\phi, 0)^2 + \max(D_{ij}^{-y}\phi, -D_{ij}^{+y}\phi, 0)^2} = \frac{1}{c_{ij}},$$

where $D_{ij}^{+x}\phi = (\phi_{i+1,j} - \phi_{ij})/\Delta x$, $D_{ij}^{-x}\phi = (\phi_{ij} - \phi_{i-1,j})/\Delta x$, etc. The extension to three spatial dimensions is straightforward. A simple method to solve this set of equations would be to apply a fixed point iteration. However, based on the above observations one can do better.

The causality property ensures that the solution of the upwind difference scheme can be computed in one node at a time by applying (16) to the unknowns $\{\phi_{ij}\}$ in the order of increasing values of ϕ_{ij} . In other words, we can compute the solution of (16) in N operations if we can determine an optimal order for the unknowns. This optimal order is characterized by increasing values of ϕ_{ij} . The fast-marching method hinges on this idea to construct the solution as an advancing front. Introducing a narrow band strategy, the solution can be computed in $\mathcal{O}(N \log N)$ operations, where $\log N$ is the cost of the ordering. For a throughout description of the fast-marching method and many of its applications, we refer to the books [11, 15].

In [7], the apparent similarities between (6) and (14) led the authors to apply the fast-marching method to (6). To justify this, the arrival-time equation was written as

$$|\nabla T| = \frac{|\nabla T|}{\mathbf{v}(\mathbf{x}) \cdot \nabla T} = \frac{1}{F(\mathbf{x})}.$$

This reformulation hides an important fact: Whereas the speed function c in (14) is isotropic, i.e., independent of orientation, F depends on T and is therefore anisotropic. Furthermore, F is not bounded away from zero and ∇T can be almost perpendicular to \mathbf{v} . When that is the case, the computed value T_k of (11) may be smaller than $\max(T_i, T_j)$. Thus, the causality property does not in general hold for the discretization of (6), and we cannot assume that the upwind scheme is

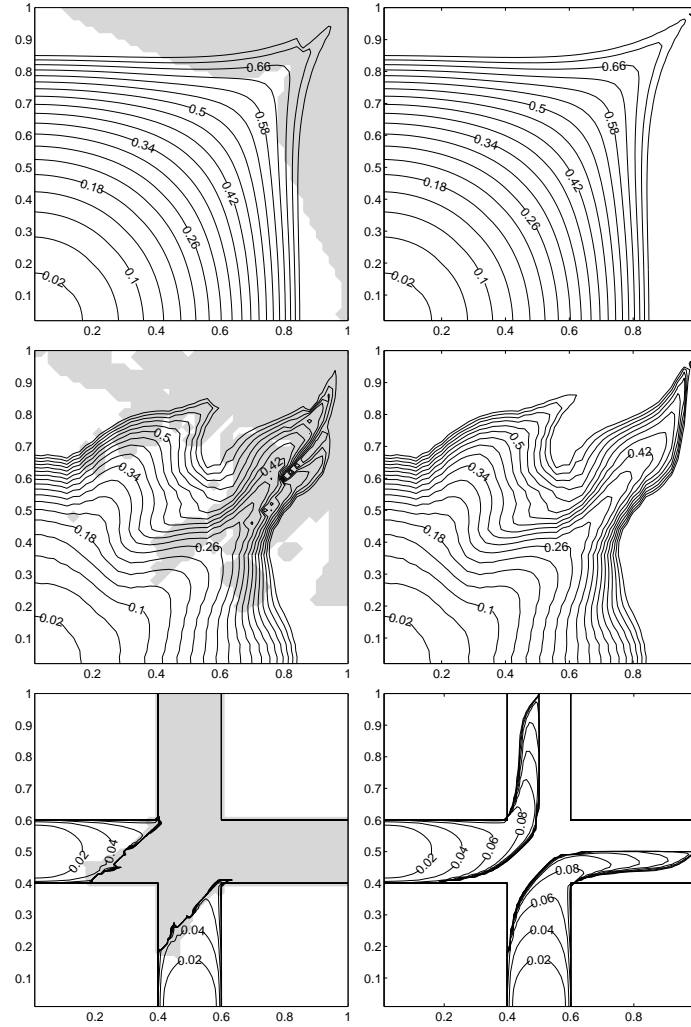


FIGURE 6. Arrival times computed with the fast-marching-like method (left) and the direct solution of (12) (right). From top to bottom, the plots correspond to Examples 1–3. The shaded region corresponds to nodes where the causality property (15) is violated.

decoupled by arranging the unknowns in the order of increasing T . Notice that the arrival-time equation (6) *has* a causality principle along streamlines, but not across streamlines. This rules out the use of the fast-marching method to solve (12). On reasonably smooth velocity fields, this defect in [7] does not severely affect the fast-marching solution of (12) as the results in [7] show. However, as the next example shows, for non-smooth velocities, the error is significant.

Example 4. We apply the fast-marching-like method in [7] to the three test cases in Examples 1 to 3 in Section 2.1. If the method is correct, we would expect to get identical results as with a direct solution of the linear system (12).

A contour plot of the arrival times computed with the fast-marching-like method, together with a direct solution of the linear system (12), is shown in Figure 6 for the three cases. As one can see, the method produces wrong solutions for all three tests, and the errors become more pronounced for the heterogeneous and layered cases in Examples 2 and 3. \square

The reason for these errors is, as has been pointed out, that the *causality* that the fast-marching method is based on, does not hold for the discretised arrival-time equation (6). The causality breaks down when the fluid velocity is nearly perpendicular to ∇T . When this happens, the nodes i and j in the difference formula (11) of T_k may have larger arrival time than T_k . Thus, T_k may be

computed based on values T_i and T_j that are not yet correct. To illustrate this in Figure 6, we have shaded the regions where the causality principle (15) is not fulfilled, i.e., where $T_k < \max(T_i, T_j)$.

These problems went unnoticed in [7, 8] due to an unforeseen side-effect of a trick used to speed up the computations of saturations. In [7] the saturation computation was viewed as the numerical evolution of a level set. The authors observed that in order to compute the saturation at a given point in time t , one only needs to compute the arrival times for nodes that have been passed by the saturation front; that is, nodes that have time-of-flight less than or equal to $T_{max} = t \max f'$. For the other (unflooded) nodes, the saturation is given by the initial saturation distribution. Thus, some nodes were left undefined in the algorithm (as they were not needed). The algorithm in [7] filled these undefined nodes with a value $T_{max} + \epsilon$, where ϵ is an arbitrary small number. If the flooded nodes had fulfilled the causality principle, that is, had they only depended on values with lesser T , this fill-inn would have had no effect. However, although the causality principle is true for the *continuous* problem, the discretization stencil frequently involves undefined nodes corresponding to larger time-of-flights, as seen in Figure 6.

The effect of the fill-in is that the magnitudes of the largest (uncomputed) arrival times are significantly reduced. This corresponds almost to a one-sided interpolation of $T(\mathbf{y})$ from the “correct” side. Therefore the results of the example in Section 5.3 of [7] look seemingly reasonable. However, the results reported in [7] are *not* solutions of the linear system (12).

3. AN IMPROVED METHOD

The question remains: Is it possible to improve the method described in [7]? The answer is yes and no. We have been able to construct a fast(er) advancing-front method for (12). This solver is discussed in the current section. On the other hand, it is harder to reduce the difficulty related to the finite-difference scheme (11) by making corrections to the interpolation procedure (11) since the time-of-flight can have large jumps. Therefore, we have implemented a method that circumvents the problem by tracking longer streamlines ψ if the solution seems to be non-smooth. We describe this scheme in Section 3.2

3.1. Advancing front solution. To improve the results from the previous sections, we have implemented a different strategy for solving (6). First, we have replaced the streamline segment ψ with a more accurate streamline approximation due to Pollock [12]. This approximation ensures that no two streamline fragments intersect, i.e., that we have a consistent approximation of streamlines.

Since the upwind scheme (11) can be constructed a priori, what remains is a system of linear equations. We know that the velocity field \mathbf{v} is divergence free and irrotational. It follows that no nodes are mutually dependent. For instance, if T_k is computed from T_i and T_j , then neither T_i nor T_j will depend on T_k . From this we can deduce that it will be possible to compute the solution T one node at a time; that is, there must exist a reordering of the unknowns that renders the system of linear equations (12) triangular.

If we can determine this ordering, we can construct the solution of (12) directly. We can, in fact, do this by traversing the nodes in the Cartesian grid Λ as an advancing front. Assume we know the correct time-of-flight in a subset $\mathcal{A} \subset \Lambda$ of nodes. Initially, this set is the set of nodes lying on the Dirichlet boundary Γ^+ . Then some nodes will depend on nodes in \mathcal{A} ; we call this set of nodes \mathcal{F} . In \mathcal{F} there are nodes that depend *only* on nodes in \mathcal{A} . If we apply (11) to these nodes, we get the correct solution of (12), and we can consider the values in these nodes known. This is the basic idea that we apply to construct the solution in N steps.

To be more precise, in each step of the algorithm we pick a node $k \in \mathcal{F}$ that depends only on nodes in \mathcal{A} through (11), that is, if T_k depends on T_i and T_j in (11), then $\{i, j\} \subseteq \mathcal{A}$. Thus, if we apply (11) to node k , we can be certain that the computed value T_k is the k 'th component of the solution of (12). To update \mathcal{F} and \mathcal{A} we then move k from \mathcal{F} to \mathcal{A} and add to \mathcal{F} any nodes that depend on k . If we proceed in this manner, we can construct the solution in all of Λ in $\mathcal{O}(N)$ operations.

In a sense, this advancing front approach is a combination of a the reordering algorithm and a back-substitution. Note that the work required for this approach is considerably reduced compared

to the fast-marching-like algorithm in [7] since no sorting is required. The algorithm is given in pseudo-code below:

```

1:  $\mathcal{A} = \{\text{Nodes on the boundary } \Gamma^+\}$ 
2:  $\mathcal{F} = \{\text{All nodes in } \mathcal{A}^C \text{ that depend on nodes in } \mathcal{A}\}$ 
3: repeat
4:   repeat
5:     pick some (random)  $k \in \mathcal{F}$ 
6:      $\mathcal{S} = \{\text{all nodes in stencil of } k\}$ 
7:   until  $\mathcal{S} \subseteq \mathcal{A}$ 
8:    $\mathcal{F} = \mathcal{F} \setminus \{k\}$ 
9:    $\mathcal{A} = \mathcal{A} \cup \{k\}$ 
10:  apply (11) to  $T_k$ 
11:   $\mathcal{F} = \mathcal{F} \cup \{\text{nodes that depend on } k\}$ 
12: until  $\mathcal{F} = \emptyset$ 

```

This algorithm will terminate if there are sufficient boundary data for the problem at hand and the velocity field has zero rotation. At every step in the algorithm, \mathcal{A} is the set of nodes for which the solution is computed, while unknowns in \mathcal{A}^C have not been computed yet. The set $\mathcal{F} \subseteq \mathcal{A}^C$ depends on \mathcal{A} through (11). By construction, this algorithm computes the correct solution of (12) in N operations. The reason for this is that the “pick” operation in line 5 can be performed in $\mathcal{O}(1)$ operations. Note also that if we order the unknowns in the order that they are processed in the outer loop, we get a triangular system.

Example 5. Let us recompute the solution of (12) for Example 3 with the advancing front approach instead of the iterative linear solver. On all three grids, the advancing front approach gives identical results with no discrepancy in the sup norm. \square

3.2. Longer streamline fragments. As we have seen, the advancing front algorithm of the previous section computes the correct solution to the linear system (12). The remaining problem related to the discretisation error is not as easy to cure. This problem is essentially a problem of too low grid resolution. Our attempt to improve the method is based on adaptivity: We wish to trace streamline segments ψ backward until a sufficiently accurate interpolation can be made. Since the time-of-flight is actually an integrated quantity, the solution does not only depend on local properties of the medium, but also on properties ‘upstream’ along a streamline. Therefore T can have large jumps transverse to the streamlines; over one grid block T can be discontinuous to grid resolution for discontinuous permeabilities. Since we have no precise *a-posteriori* error measure for the difference scheme (11) the adaptive approach must necessarily involve some ad hoc features.

The proposed algorithm is essentially the same as the advancing front approach, with one exception: For each unknown that is to be computed, we trace a streamline backwards until the interpolation seems to be reasonably accurate. A very rough indicator of whether the discretization error is large or small is to compare ΔT_k of (10) to $|T_i - T_j|$ in (11). If $\Delta T_k > C|T_i - T_j|$ for some fixed C we accept the the update formula based on the nodes i and j . Otherwise, we trace the streamline ψ_k further back. We emphasize that this indicator for interpolation error is ad hoc and does not have a rigorous basis.

Example 6. We apply the improved method of Section 3.1 and 3.2 to our three test problems. For all the three cases we let $C = 0.1$, and, as an additional test, we also try $C = 1.0$ for the heterogeneous case.

The left column in the Figures 7, 8 and 9 show contour plots of the results from the adaptive algorithm. For extra illustration, we have shaded the region where nodes have been updated using longer streamline fragments. In the right hand column in Figures 7–9, we have plotted the difference between the solution from the adaptive scheme and the direct evaluation of (7). Note that nodes in the regions where the difference between the two solutions are greater than $\mathcal{O}(h)$ generally have large arrival times. \square

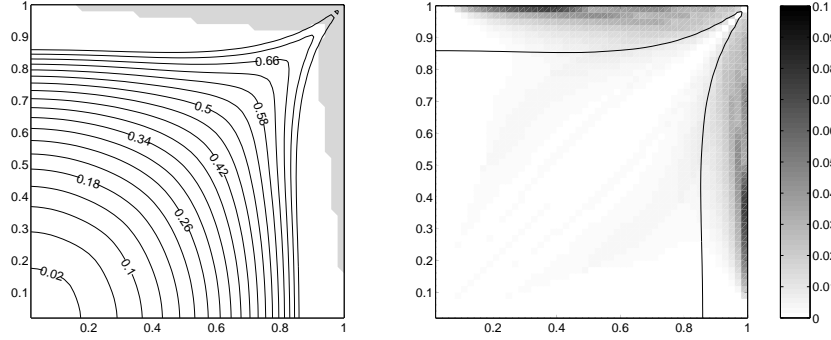


FIGURE 7. Arrival times for the homogeneous example. (Left) Solution computed with the adaptive algorithm with $C = 0.1$. Contours are drawn for $T \leq 0.74$. The shaded region shows nodes that has been updated by using longer streamline fragments. (Right) Difference between the adaptive scheme and the direct evaluation of (7), together with a plot of the contour $T = 0.74$ for the direct evaluation. The mesh spacing is $h = 0.02$.

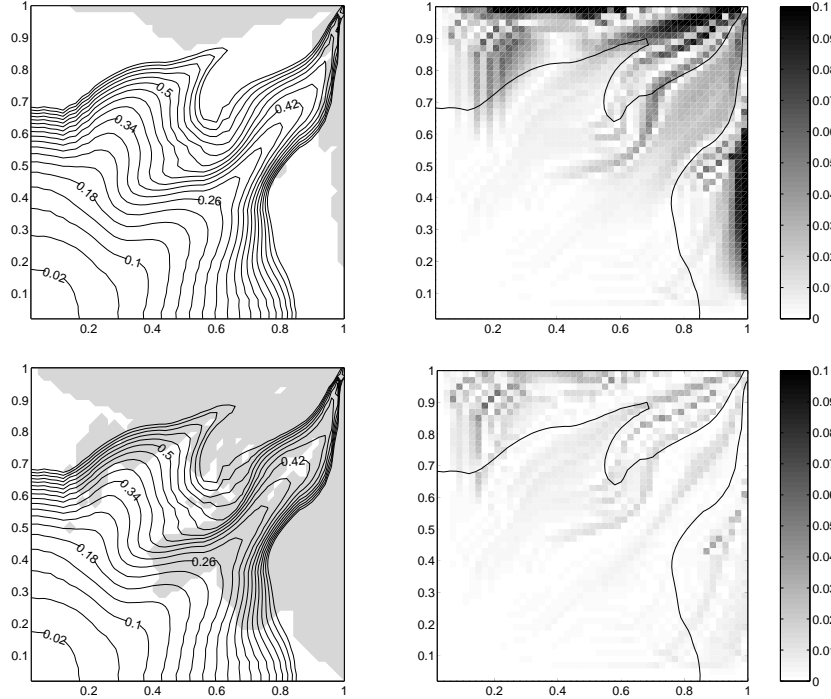


FIGURE 8. Arrival times for the heterogeneous example. (left column) Solutions computed with the adaptive algorithm. Contours are drawn for $T \leq 0.70$. The shaded region shows nodes that has been updated by using longer streamline fragments. (right column) Difference between the adaptive scheme and the direct evaluation of (7), together with a plot of the contour $T = 0.70$ for the direct evaluation. The mesh spacing $h = 0.02$. The threshold parameters are $C = 0.1$ (upper row) and $C = 1.0$ (lower row).

Example 6 demonstrates that the adaptive scheme gives reasonably good results, also for the channel problem and the heterogeneous medium. However, the added computational cost of the adaptive algorithm is quite high, and we have observed that this added work is spent mostly in areas where ∇T is almost perpendicular to \mathbf{v} . In other words, the work is spent in regions where there is a large shear in the flow. These areas seem to be of little significance in the early stages

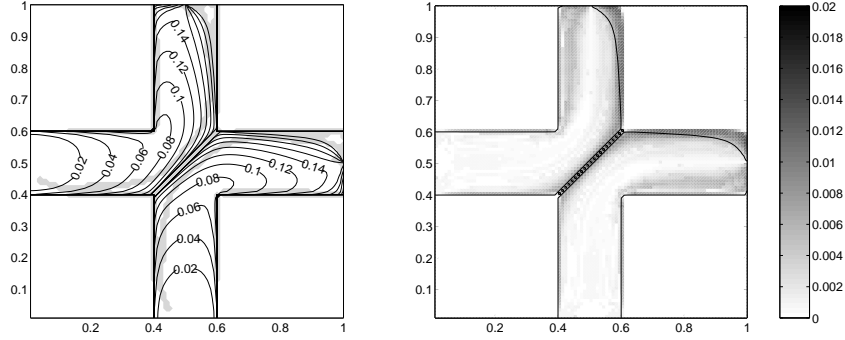


FIGURE 9. Arrival times for the channel example. (left) Solution computed with the adaptive algorithm with $C = 0.1$. Contours are drawn for $T \leq 0.24$. The shaded region shows nodes that has been updated by using longer streamline fragments. (right) Difference between the adaptive scheme and the direct evaluation of (7) together with a plot of the contour $T = 0.24$ for the direct evaluation. The mesh spacing $h = 0.01$.

of a reservoir as the nodes are flooded long after water breakthrough. Another problem with this method is that the parameter C is hard to specify in advance. For a specific example, one may adjust the accuracy by adjusting the parameter, but there is no guarantee that this C -value will yield good results for a different example. Thus, the C -parameter may have to be tuned on a case-to-case basis. To be practical, the method must include a more precise error measure that is cheap to compute.

4. DISCUSSIONS AND RELATIONS TO STREAMLINE METHODS

In this paper we have carefully investigated and extended the fast saturation solver presented in [7]. In our analysis, we have interpreted the method in terms of the time-of-flight formalism. It is evident from this interpretation that there is a clear relation between streamline methods and our marching method(s). Therefore, it is natural to compare the possibilities and limitations of the marching methods presented herein with those of streamline methods.

The splitting technique applied in [7] and in this paper is limited to initial boundary value problems with an inherent (radial) symmetry. This symmetry ensures that the three-dimensional initial-boundary-value problem for the saturation equation can be decomposed into a single one-dimensional conservation law for the behavior along any streamline and the computation of an appropriate coordinate transform. This coordinate transform allows the mapping of the one-dimensional solution into three-dimensional space. The approach presented herein differs from streamline methods mainly in the calculation of the coordinate transform. This coordinate transform is more commonly interpreted as time-of-flight in literature on streamline methods. In the current paper, the coordinate transform is found by solving the arrival-time equation. As demonstrated, it is possible to construct special purpose solvers for this equation, which seems to be fast and reliable for permeability fields that are either homogeneous or vary relatively smoothly. However, for high contrast layered media where the flow is parallel to the layers, the marching method produces inaccurate solutions. The reason for this is that the time-of-flight is less smooth for these problems resulting in large discretisation errors.

As these problems are caused by discretisation errors, we have proposed an adaptive scheme that changes the difference formula in the presence of large variations in time-of-flight. The adaptivity reduces the problems with the discretisation errors by avoiding interpolation in regions where an interpolation will result in large discretisation errors, typically where the gradients in time-of-flight are nearly perpendicular to the fluid velocity. The adaptive scheme seems to be able to better compute time-of-flight than a fixed difference scheme, but the adaptivity is quite expensive. Most of the extra work is spent in regions with high arrival times due to the rough error measure used here. Because of the extra work, the speed of the method is reduced, and is probably comparable

to, if not slower than other methods like streamline methods. A more accurate error measure is therefore needed to make the method practical.

Streamline methods share many properties with the advancing front method and the method presented in [7], but do not suffer from the same limitations. To be more specific, streamline methods map one-dimensional solutions back to three-dimensional space by computing particle paths (streamlines). These computations make streamline methods more expensive than the fast finite-difference solver presented in Section 3.1. However, the computation of streamlines is well-behaved for all the examples presented in the paper, and streamline methods will therefore accurately the time-of-flight in all examples. Furthermore, the tracking of individual particle paths leaves room for more general problems where a different one-dimensional solution must be computed for each path. Thus, it is fair to say that methods based on streamline integration are more robust and general than the approach taken here.

ACKNOWLEDGMENT

The research of Karlsen, Lie, and Natvig was supported by the BeMatA program of the Research Council of Norway. Karlsen was also supported in part by the European network HYKE, funded by the EC as contract HPRN-CT-2002-00282

REFERENCES

- [1] K. Aziz and A. Settari. *Petroleum reservoir simulation*. Elsevier Applied Science Publishers, Essex, England, 1979.
- [2] I. Berre, H. K. Dahle, K. H. Karlsen, K.-A. Lie, and J. R. Natvig. Time-of-flight + fast marching + transport collapse: an alternative to streamlines for two-phase porous media flow with capillary forces? *Proc. CMWR XIV*, Volume 2, Elsevier, Amsterdam (2002), pages 995–1002.
- [3] G. Chavent and J. Jaffre. *Mathematical models and finite elements for reservoir simulation*, vol. 17 of *Studies in mathematics and its applications*. North Holland, Amsterdam, 1986.
- [4] A. Datta-Gupta and M. J. King. A semi-analytic approach to tracer flow modeling in heterogenous permeable media. *Advances in Water Resources*. Res.,18(1):9–24, 1995.
- [5] M. S. Espedal and K. H. Karlsen. Numerical solution of reservoir flow models based on large time step operator splitting algorithms. In volume 1734 of *Lecture Notes in Mathematics*, pages 9–77. Springer, Berlin, 2000.
- [6] J. Helmsen, E. G. Puckett, P. Colella, and M. Dorr. Two new methods for simulating photolithography development in 3d. *Proc. SPIE*, 2726:253–261, 1996.
- [7] K. H. Karlsen, K.-A. Lie, and N. H. Risebro. A fast marching method for reservoir simulation. *Comp. Geo.*, 4(2):185–206, 2000.
- [8] K. H. Karlsen, K.-A. Lie, J. R. Natvig, and N. H. Risebro. A fast marching method for 3D reservoir simulation. *Proc. NSCM-13*, Mechanics and Applied Mathematics Series, No. 7, pp. 147–150, University of Oslo, Norway, October 2000.
- [9] M. J. King and A. D. Datta-Gupta. Streamline simulation: a current perspective. *In Situ (Special Issue on Reservoir Simulation)*, 22(1):91–140, 1998.
- [10] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.
- [11] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer Verlag, New York, 2002.
- [12] D. W. Pollock. Semianalytical computation of path lines for finite difference models. *Ground Water* 26(6):743–750, 1988.
- [13] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci. U.S.A.*, 93(4):1591–1595, 1996.
- [14] J. A. Sethian. Fast marching level set methods for three-dimensional photolithography development. *Proc. SPIE*, 2726:261–272, 1996.
- [15] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, Cambridge, 1999.
- [16] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In *Proceedings of IEEE 33rd Conference on Decision and Control, Lake Buena Vista, Fl, Dec. (1994)*, pages 1368–1373.
- [17] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.